Image Recognition for Flower Species Identification

1 Introduction

Flower classification is required for botany survey, data management and taxonomy applications (Chi, 2003). Recognition of species, shape, colour, texture and scale, combined with position, intra-species variation and image background complexity, complicates feature selection and classification (Cibuk et al., 2019).

CNNs classify images without image segmentation (Simonyan & Zisserman, 2014; Song et al., 2016). High accuracy (>95%) with and without segmentation on varying species number has been achieved (Nilsback & Zisserman, 2008; Razavian et al., 2014; Prasad et al., 2017; Xia et al., 2017; Hiary et al., 2018; Sai et al., 2022). Cibuk et al. (2019) used pre-trained AlexNet and VGG16 models with a SVM classifier for entire image classification demonstrating relative simplicity and excellent accuracy (Cibuk et al., 2019). Classification of five flower species with a four convolutional layer CNN achieved 74% accuracy on a small sample of 100 images (FatihahSahidan et al., 2019). A four convolutional layer CNN for 26 species classification of 1300 images achieved 95% accuracy with reduced classification performance for species with similar shape, texture and colour (Omer et al., 2020).

This work contributes a novel CNN model, optimising for best accuracy and performance for five flower species classification without image segmentation, on a larger dataset.

2 Methods

The code and environment versions are provided in the Jupyter notebook (CNN_flowers_AI_SarahRogers.ipynb).

The tf_flowers dataset includes five flower species, daisy, dandelion, rose, sunflower and tulip, imported using image_dataset_from_directory (Tensorflow, 2023m). Image size is 256 x 256 pixels.

Species numerical classifiers are:

- Daisy = 0
- Dandelion = 1
- Rose = 2
- Sunflowers = 3
- Tulip = 4

2.1 CNN Model

2.1.1 Model architecture

The final optimised CNN architecture is detailed in Appendix 6, Figure 1. The traditional architecture was partly based on AlexNet and VGG16 models, previously used for flower classification (Cibuk et al., 2019). The model has five convolutional layers, with a similar architecture to AlexNet, with max pooling, dropout, batch normalisation and ReLU activation (Krizhevsky et al., 2012). It has an additional convolutional layer compared to more recent CNN models for flower species classification and pooling layers between each convolutional layer (FatihahSahidan et al., 2019; Omer et al., 2020).

The operation in convolutional layers slides a kernel across the dimension of the image, extracting features (hierarchical patterns) and producing feature maps. An activation function introduces non-linearity for complex relationship learning (Chauhan et al., 2018). Pooling layers subsample important features. Dense layers generate predictions from important features from convolutional and pooling

layers (O'Shea & Nash, 2015; Lee & Song, 2019). The output layer contains Softmax for classification (Kouretas & Paliouras, 2020).

2.1.2 Optimisation

2.1.2.1 Regularisation

2.1.2.1.1 Data augmentation

Augmentation included 20-degree image rotation, 10% horizontal and vertical shifts, horizontal flipping, 10% zoom, and shear. Random transformations applied to each batch, diversified the dataset improving generalisation (Tensorflow, 2023k).

2.1.2.1.2 Dropout

Randomly dropping neurons ignored during model training, reduces reliance on specific features and neurons introducing variability (Tensorflow, 2023c). Dropout rate of 20-50% is explored alone and in combination with additional regularisation.

2.1.2.1.3 Batch normalisation

Batch normalisation, adjusts mean and standard deviation, promoting stable convergence and faster training. Scale and shift creates noise, providing flexibility and reducing internal covariate shift (Dive into deep learning, n .d.). Momentum is investigated at 0.9 and 0.95, with typical values from 0.9 to 0.99 (Ioffe & Szegedy, 2015).

2.1.2.1.4 Early stopping

Early stopping prevents overfitting by monitoring accuracy or loss of the validation data. If performance stalls or declines, the training halts before overfitting, determined by the patience (7 in this work) (Tensorflow, 2023a). Number of epochs is adjusted to ensure training convergence with and without early stopping.

2.1.2.1.5 L1 and L2

L1 and L2 penalises loss proportional to the absolute and squared values of the weights, respectively. L1 emphasises important features by weighting less important features to zero. L2 evens the weights with no single feature influencing training. Elastic net provides a balance between L2 sparsity and L2 weighting reduction (Neptune.ai, 2023; Tensorflow, 2023l). Strength is tuned between 0.001 and 1.

2.1.2.2 Hyperparameter tuning

2.1.2.2.1 Convolutional layers

2.1.2.2.1.1 Kernel size

Smaller kernel sizes can capture finer details and larger kernels capture global features. Smaller kernels are computationally efficient, and it is common to have multiple convolutional layers with small kernels stacked in the network. Smaller kernels typically have smaller stride (discussed in section 2.1.2.2.1.2), increasing stride when overlap becomes an issue with larger kernels (Chansong & Supratid, 2021). Kernel sizes ((3, 3), (5, 5)) are investigated.

2.1.2.2.1.2 Stride and padding

Stride (the step size the kernel takes across the feature map) and padding (additional pixels at image edges capturing all possible features) changes the dimensions of the feature map. Smaller strides capture finer details (Dive into deep learning, n. d.). Padding and stride ((2, 2), (3, 3)) are investigated.

2.1.2.2.1.3 Activation function

Complex patterns are captured by introducing non-linearity into the model with the activation function. ReLU activation is commonly used due to its simplicity, but SeLU and eLU are also explored (Nair & Hinton, 2010; Tensorflow, 2023d; 2023g; 2023f; Keras, n. d.-b).

2.1.2.2.2 Batch size, learning rate and pooling

32, 64 and 128 batch size on weight updating is determined. Smaller batch sizes may introduce more noise aiding generalisation with more frequent weight updates. Larger batches facilitates a smoother gradient, potentially requiring increased learning rate, increasing convergence rate (Kandel & Castelli, 2020).

Learning rate for Adam optimiser, is 0.01 and 0.001. Alternative optimisers are not considered (Tensorflow, 2023j; Adam, n. d.). Max and average pooling are considered for down-sampling (Tensorflow, 2023b; 2023e).

2.1.3 Evaluation metrics

Training and validation evaluation metrics are accuracy and loss (categorical cross-entropy) (Tensorflow, 2023h; 2023i).

Metrics for classification performance are precision (number of predicted positives that are true positives), recall (how many positive instances were captured), F1 score (balance of precision and recall) and support (number in each class), using classification report and confusion matrix visualisation (Scikit learn, n. d.-c; n. d.-b; n. d.-a).

3 Results and Discussion

The final architecture (Appendix 6.1, Figure 1) has five convolutional layers with ReLU activation, learning rate 0.001 (Adam), kernel size (3, 3), stride (3, 3), and 64, 128, 128, 256 and 512 filters, respectively. Increasing filters through the model is a typical approach in established models like VGGNet (Simonyan & Zisserman, 2014). A pooling (MaxPooling, pool size (2, 2)) and dropout layer (rate 0.3) layer follows each convolutional layer. Padding was added to all convolutional and pooling layers. A final dense layer (512 filters, ReLU activation) precedes the Softmax output layer for flower species classification.

The final architecture for batch size 32, ran for 40 epochs with test set accuracy of 74% (Appendix 6.1, Figure 2). Performance was comparable to previous research on reduced image numbers, and performed well on the larger data set (FatihahSahidan et al., 2019). Reasonable classification was achieved, except tulips and roses are misclassified in some cases (Appendix 6.1, Figure 3), indicated by reduced precision and f1 score for both species, potentially due to visual similarities in shape, texture, and colour. High variation in these class images may inhibit effective learning of distinctive features.

Reducing convolutional layers from five to four led to lower accuracy and classification performance in distinguishing roses and tulips, as well as misclassifying both as dandelions (Appendix 6.1, Figures 4, 5). Slight overfitting is evident with a small gap in loss and accuracy between training and validation data. Four-layer models in prior flower classification studies, using smaller datasets, achieved similar or higher accuracy (FatihahSahidan et al., 2019; Omer et al., 2020). Considering Occam's razor, the complexity of five-layers proves more suitable for analyzing increased numbers of flower images and was chosen for further investigation of optimal hyperparameters and regularisation to increase accuracy and limit overfitting (New Scientist, n. d.).

Regularisation methods explored for the five-layer model were batch normalization, L1 and L2, dropout and early stopping (detailed in Appendix 6.2, Table 1). Optimal dropout rate was 0.3, with 0.5 significantly decreasing accuracy, performance and introducing significant overfitting (Appendix 6.1, Figures 6, 7) (Yang & Yang, 2018). Batch normalisation with a momentum of 0.90 and 0.95 did not improve accuracy or classification performance, with overfitting present (Appendix 6.1, Figures 8-11). Combined with L1 and L2 (elastic net) at strength 0.001, 0.01, 0.1 and 1 accuracy was improved at lower strengths but overfitting was evident and classification performance was below that of models without L1 and L2 (Appendix 6.1, Figures 12-19).

Early stopping was explored (patience 7) but failed to halt training for 40 epochs. Further work is required to optimise patience and min_delta for validation data noise (Prechelt, 2012).

The hyperparameter tuning, test accuracy, and details of over-fitting and misclassification are detailed in Appendix 6.2, Table 2. Batch size, stride and padding had the strongest influence on accuracy and performance. A lower batch size of 32, increased accuracy and improved classification performance (Appendix 6.1, Figures 20, 21). Overfitting was overcome by the addition of padding and stride (3, 3) without detrimental loss in accuracy (Appendix 6.1, Figures 22, 23). Increasing Adam learning rate to 0.01 from default (0.001) increased overfitting in combination with regularisation techniques above (Appendix 6.1, Figures 24, 25). No accuracy improvement was achieved with batch normalization and L1 and L2 regularisation (0.001) for average pooling (Appendix 6.1, Figures 26, 27), or alternative activation functions, SeLU and eLU (Appendix 6.1, Figures 28-31). Increasing kernel size to (5,5) decreased accuracy and classification performance (Appendix 6.1, Figures 32, 33).

In the absence of early stopping and additional regularisation techniques besides dropout, determining optimal training epochs ensured generalisation without overfitting, a trade-off with accuracy and classification performance.

4 Conclusion

A five convolutional layer CNN achieved the best accuracy and classification of the flower image dataset, with comparable performance to literature on a larger number of images. Data augmentation and dropout regularisation improved the generalisation of the model to test data. Stride, padding and batch size contributed most significantly to accuracy and classification performance. The model underperformed for classification of tulips and roses.

Further work could explore L1 or L2 regularisation alone, or in combination with differing strengths. More efficient optimisation could be achieved with random search and Keras tuner (Keras, n. d.-a). Transfer learning using pre-trained models or ensemble learning (bagging or boosting) are alternative approaches (Narvekar & Rao, 2020; Wang et al., 2022).

Word count 1482

5 References

Adam (n. d.) *Layer activation functions.* Available online: https://keras.io/api/optimizers/adam/ [Accessed 16/12/2023].

Chansong, D. & Supratid, S. (2021) Impacts of kernel size on different resized images in object recognition based on convolutional neural network, *IEEE 9th International Electrical Engineering Congress.* Pattaya, Thailand, 10-12 March 2021. 448-451.

Chauhan, R., Ghanshala, K. K. & Joshi, R. C. (2018) Convolutional neural network (CNN) for image detection and recognition, *IEEE 1st International Conference on Secure Cyber Computing and Communications.* Jalandhar, India, 15-17 December 2017. 278-282.

Chi, Z. (2003) Data management for live plant identification, in Feng, D. D., Siu, W.-C. & Zhang, H.-J. (eds), *Multimedia information retrieval and management: technological fundamentals and applications*. Berlin, Heidelberg: Springer Nature, 432-457.

Cibuk, M., Budak, U., Guo, Y. H., Ince, M. C. & Sengur, A. (2019) Efficient deep features selections and classification for flower species recognition. *Measurement*, 137, 7-13.

Dive into deep learning (n .d.) *Batch normalization.* Available online: https://d2l.ai/chapter_convolutional-modern/batch-norm.html [Accessed 16/12/2023].

Dive into deep learning (n. d.) *Padding and stride.* Available online: https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html [Accessed 16/12/2023].

FatihahSahidan, N., Khairi Juha, A., Mohammad, N. & Ibrahim, Z. (2019) Flower and leaf recognition for plant identification using convolutional neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, 16(12), 2502-4752.

Hiary, H., Saadeh, H., Saadeh, M. & Yaqub, M. (2018) Flower classification using deep convolutional neural networks. *Let Computer Vision*, 12(6), 855-862.

Ioffe, S. & Szegedy, C. (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift, *32nd International Conference on Machine Learning.* Lille, France, 7-9 July 2015. 448-456.

Kandel, I. & Castelli, M. (2020) The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4), 312-315.

Keras (n. d.-a) *Keras tuner.* Available online: https://keras.io/keras_tuner/ [Accessed 29/12/2023].

Keras (n. d.-b) *Layer activation functions.* Available online: https://keras.io/api/layers/activations/ [Accessed 16/12/2023].

Kouretas, I. & Paliouras, V. (2020) Hardware implementation of a Softmax-like function for deep learning. *Technologies*, 8(3), 46.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012) ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 84-90.

Lee, H. & Song, J. (2019) Introduction to convolutional neural network using Keras; an understanding from a statistician. *Communications for Statistical Applications and Methods*, 26(6), 591-610.

Nair, V. & Hinton, G. E. (2010) Rectified linear units improve restricted Boltzmann machines, *27th International Conference on Machine Learning.* Haifa, Israel, 21-24 June 2010.

Narvekar, C. & Rao, M. (2020) Flower classification using CNN and transfer learning in CNN-agriculture perspective, *IEEE 3rd international conference on intelligent sustainable systems.* Thoothukudi, India, 3-5 December 2020. 660-664.

Neptune.ai (2023) *Fighting overfitting with L1 or L2 regularization: which one is better?* Available online: https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization [Accessed 16/12/2023].

New Scientist (n. d.) *Occam's razor.* Available online: https://www.newscientist.com/definition/occams-razor/ [Accessed 29/12/2023].

Nilsback, M.-E. & Zisserman, A. (2008) Automated flower classification over a large number of classes, *IEEE 6th Indian conference on computer vision, graphics & image processing.* Bhubaneswar, India, 16-19 December 2008. 722-729.

O'Shea, K. & Nash, R. (2015) An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Omer, S. M., Hasan, R. M. & Anwer, B. N. (2020) An image dataset construction for flower recognition using convolutional neural network. *Science Journal of University of Zakho*, 8(3), 112-117.

Prasad, M. V. D., JwalaLakshmamma, B., Hari Chandana, A., Komali, K., Manoja, M. V. N., Rajesh Kumar, P., Raghava Prasad, C., Inthiyaz, S. & Sasi Kiran, P. (2017) An efficient classification of flower images with convolutional neural networks. *International Journal of Engineering & Technology*, 7(1.1), 384-391.

Prechelt, L. (2012) *Neural networks tricks of the trade: early stopping - but when?*, 2nd edition. Berlin, Heidelberg: Springer Nature.

Razavian, A., Azizpour, H. & Sullivan, J. (2014) CNN features off-the-shelf: an astounding baseline for recognition, *IEEE Conference on Computer Vision and Pattern Recognition.* Columbus, USA, 23-28 June 2014. 512-519.

Sai, A. V., Hitesh, M. S. V., Jadala, V. C., Pasupuleti, S. K., Raju, S. H. & Shameem, M. (2022) Flower identification and classification applying CNN through deep learning methodologies, *IEEE International Mobile and Embedded Technology Conference.* Noida, India, 10-11 March 2022. 173-180.

Scikit learn (n. d.-a) *Confusion matrix.* Available online: https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html [Accessed 16/12/2023].

Scikit learn (n. d.-b) *sklearn.metrics.classification_report.* Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html [Accessed 16/12/2023].

Scikit learn (n. d.-c) *sklearn.metrics.precision_recall_fscore_support.* Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html [Accessed 16/12/2023].

Simonyan, K. & Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition, *International Conference on Learning Representations*. San Diego, USA, 7-9 May 2015..

Song, G.-H., Jin, X.-G., Chen, G.-I. & Nie, Y. (2016) Two-level hierarchical feature learning for image classification. *Frontiers of Information Technology & Electronic Engineering*, 17(9), 897-906.

Tensorflow (2023a) *tf.keras.callbacks.EarlyStopping.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping [Accessed 16/12/2023].

Tensorflow (2023b) *tf.keras.layers.AveragePooling2D.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/AveragePooling2D [Accessed 16/12/2023].

Tensorflow (2023c) *tf.keras.layers.Dropout.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout [Accessed 16/12/2023].

Tensorflow (2023d) *tf.keras.layers.ELU.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/ELU [Accessed 16/12/2023].

Tensorflow (2023e) *tf.keras.layers.MaxPooling2D.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPooling2D [Accessed 16/12/2023].

Tensorflow (2023f) *tf.keras.layers.ReLU.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU [Accessed 16/12/2023].

Tensorflow (2023g) *tf.keras.layers.SeLU.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/activations/selu [Accessed 16/12/2023].

Tensorflow (2023h) *tf.keras.losses.CategoricalCrossentropy.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy [Accessed 16/12/2023].

Tensorflow (2023i) *tf.keras.metrics.Accuracy.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy [Accessed 16/12/2023].

Tensorflow (2023j) *tf.keras.optimizers.Adam.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam [Accessed 16/12/2023].

Tensorflow (2023k) *tf.keras.preprocessing.image.ImageDataGenerator.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator [Accessed 16/12/2023].

Tensorflow (2023l) *tf.keras.regularizers.Regularizer.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/Regularizer [Accessed 16/12/2023].

Tensorflow (2023m) *tf.keras.utils.image_dataset_from_directory.* Available online: https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory [Accessed 16/12/2023].

Wang, Z., Wang, K., Wang, X., Pan, S. & Qiao, X. (2022) Dynamic ensemble selection of convolutional neural networks and its application in flower classification. *International Journal of Agricultural and Biological Engineering*, 15(1), 216-223.

Xia, X., Xu, C. & Nan, B. (2017) Inception-v3 for flower classification, *IEEE 2nd International Conference on Image, Vision and Computing.* Chengdu, China, 2-4 June 2017. 783-787.

Yang, J. & Yang, G. C. (2018) Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms*, 11(3), 28.

6 Appendix

6.1 Figures

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 86, 86, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 43, 43, 64)       0
 )

 dropout (Dropout)           (None, 43, 43, 64)        0

 conv2d_1 (Conv2D)           (None, 15, 15, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 128)         0

 conv2d_2 (Conv2D)           (None, 3, 3, 128)         147584

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 128)         0

 conv2d_3 (Conv2D)           (None, 1, 1, 256)         295168

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 dropout_3 (Dropout)         (None, 1, 1, 256)         0

 conv2d_4 (Conv2D)           (None, 1, 1, 512)         1180160

 max_pooling2d_4 (MaxPooling  (None, 1, 1, 512)        0
 2D)

 dropout_4 (Dropout)         (None, 1, 1, 512)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_5 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 5)                 2565

=================================================================
Total params: 1,963,781
Trainable params: 1,963,781
Non-trainable params: 0
_________________________________________________________________
```

Figure 1. Final optimised CNN architecture

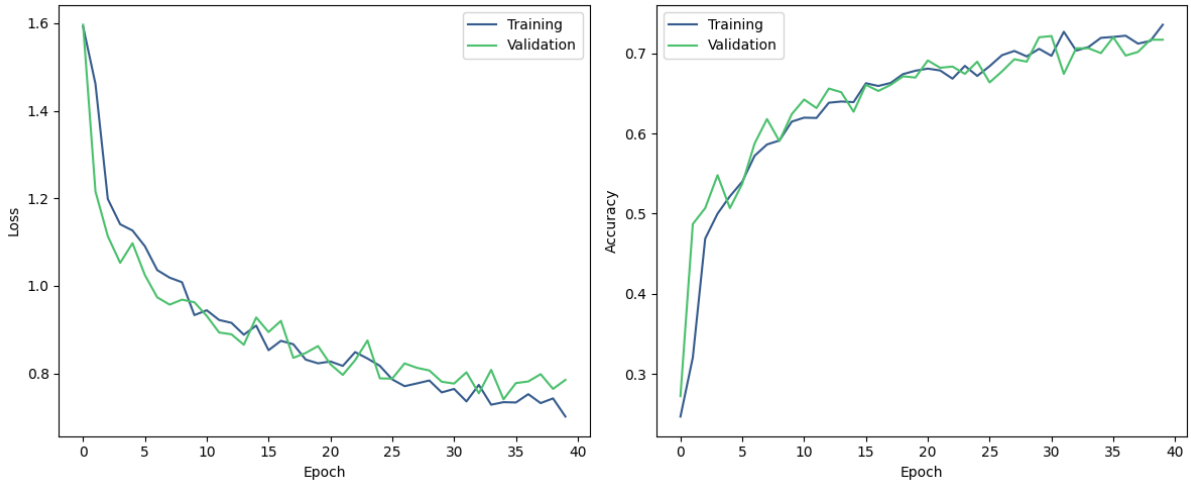

Figure 2. Final optimised CNN training and validation accuracy and loss

	precision	recall	f1-score	support
0	0.74	0.78	0.76	50
1	0.76	0.81	0.78	91
2	0.64	0.69	0.66	68
3	0.84	0.73	0.78	70
4	0.73	0.67	0.70	86
accuracy			0.74	365
macro avg	0.74	0.74	0.74	365
weighted avg	0.74	0.74	0.74	365

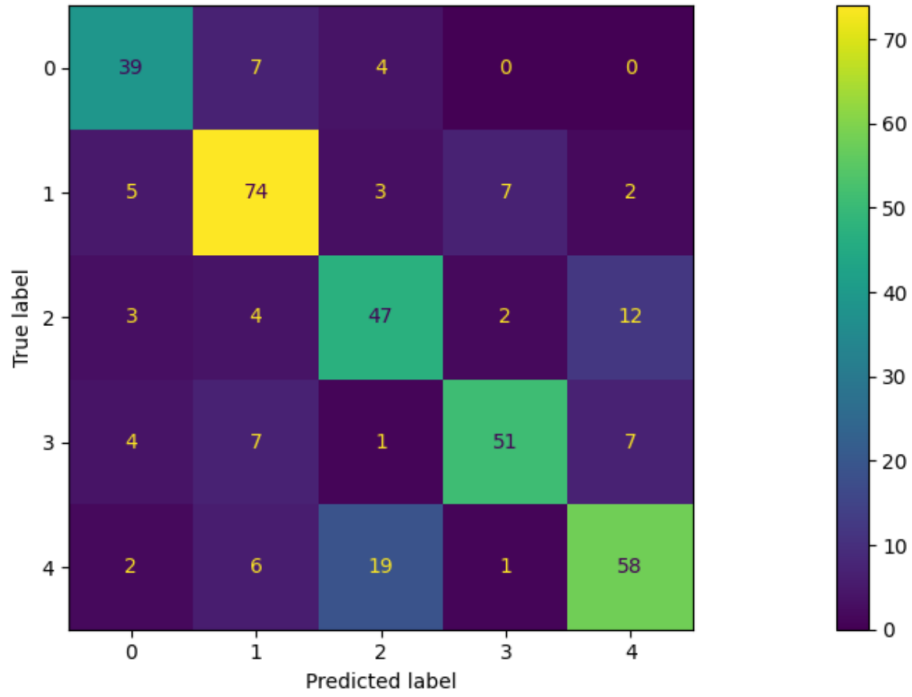

Figure 3. Final optimised CNN confusion matrix and classification report

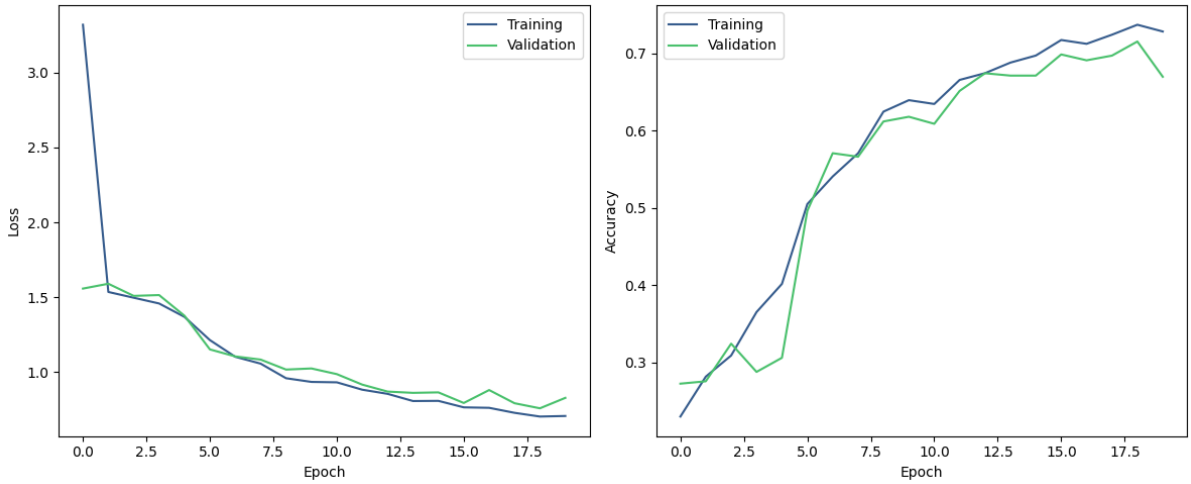

Figure 4. Four-layer CNN training and validation accuracy and loss

	precision	recall	f1-score	support
0	0.78	0.80	0.79	50
1	0.63	0.91	0.74	91
2	0.83	0.29	0.43	68
3	0.83	0.84	0.84	70
4	0.66	0.66	0.66	86
accuracy			0.71	365
macro avg	0.75	0.70	0.69	365
weighted avg	0.73	0.71	0.69	365

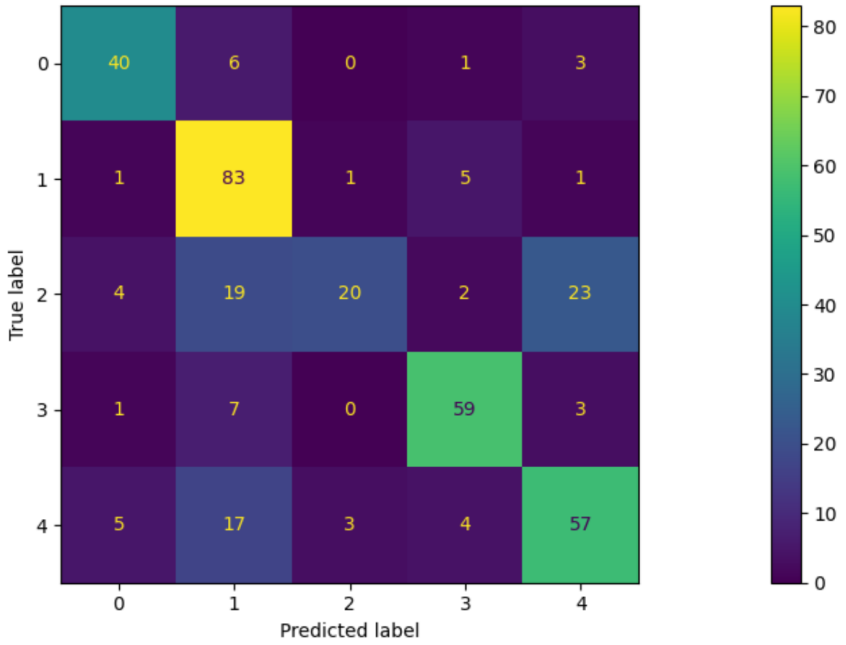

Figure 5. Four-layer CNN confusion matrix and classification report

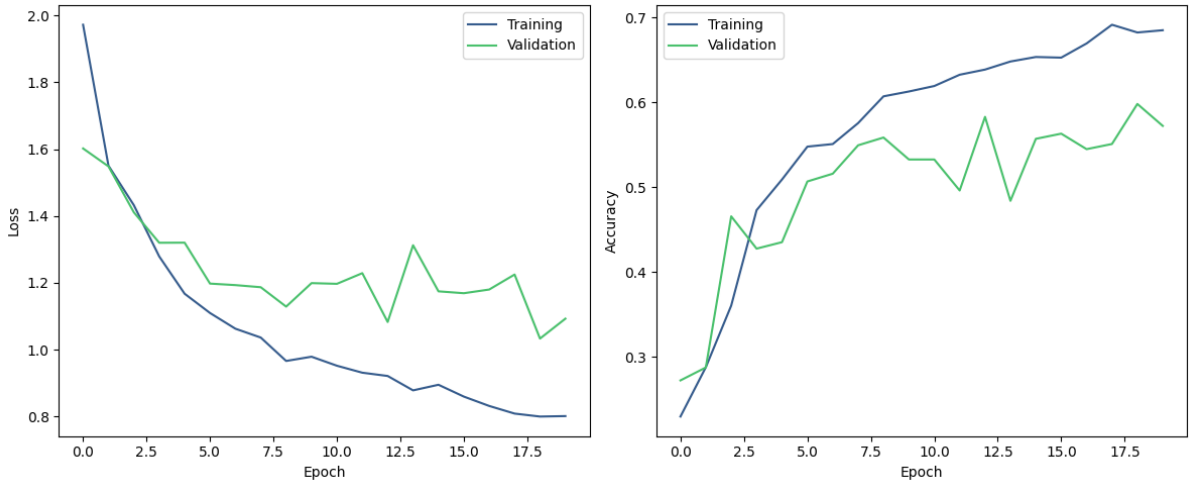

Figure 6. Training and validation accuracy and loss for 0.5 dropout rate on five-layer CNN

	precision	recall	f1-score	support
0	0.61	0.90	0.73	50
1	0.51	0.87	0.64	91
2	0.39	0.25	0.30	68
3	0.76	0.79	0.77	70
4	0.86	0.21	0.34	86
accuracy			0.59	365
macro avg	0.63	0.60	0.56	365
weighted avg	0.63	0.59	0.54	365

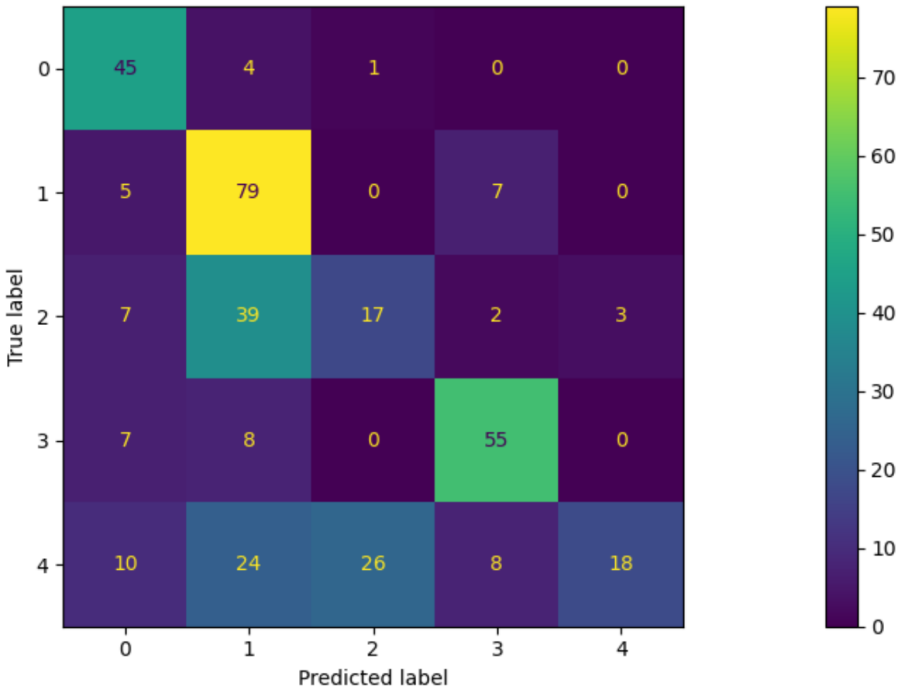

Figure 7. Confusion matrix and classification report for 0.5 dropout rate

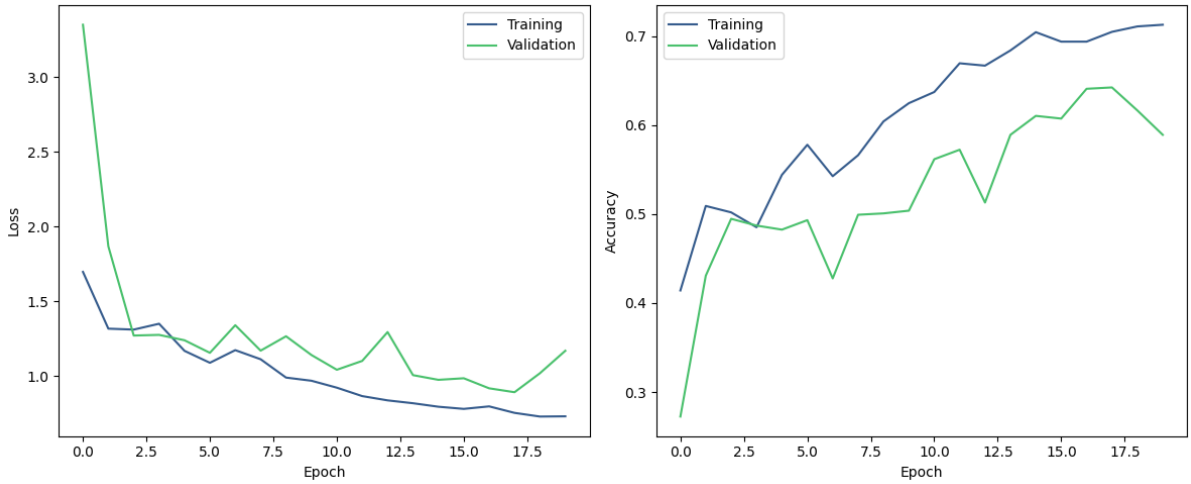

Figure 8. Training and validation accuracy and loss for batch normalisation, momentum 0.9

	precision	recall	f1-score	support
0	0.66	0.82	0.73	50
1	0.79	0.81	0.80	91
2	0.66	0.49	0.56	68
3	0.84	0.84	0.84	70
4	0.70	0.72	0.71	86
accuracy			0.74	365
macro avg	0.73	0.74	0.73	365
weighted avg	0.74	0.74	0.73	365

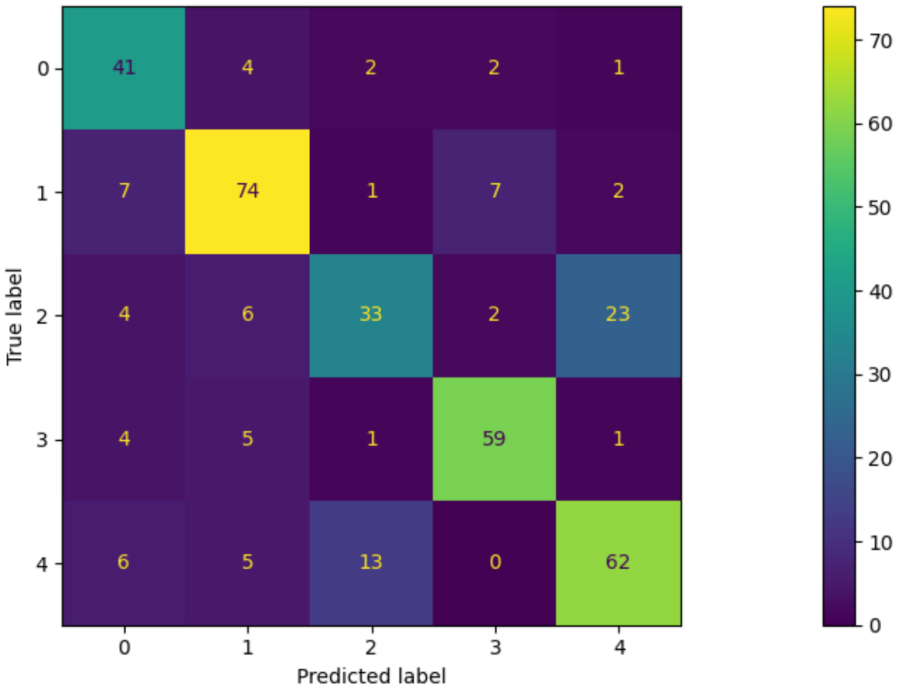

Figure 9. Confusion matrix and classification report for batch normalisation, momentum 0.9

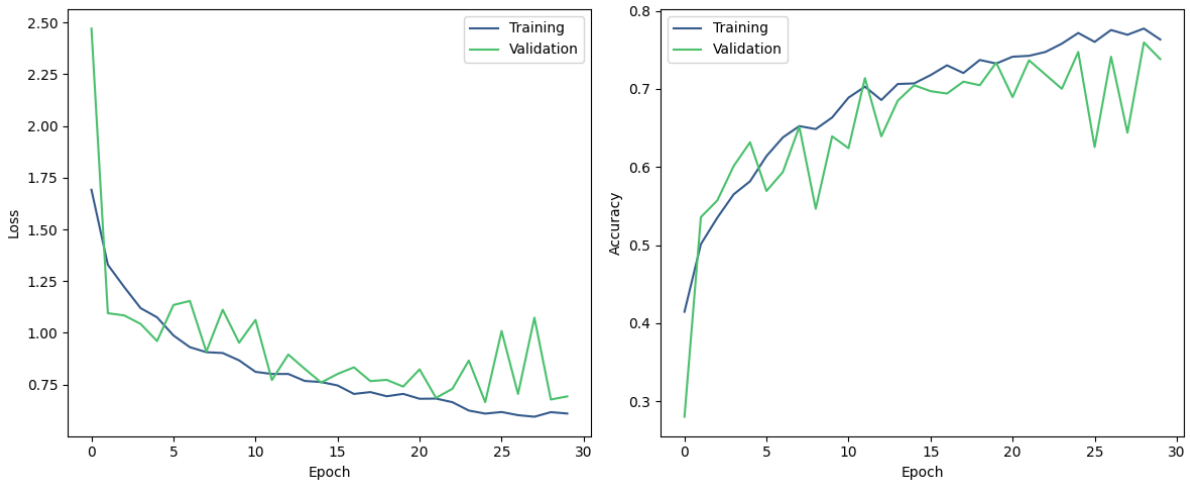

Figure 10. Training and validation accuracy and loss for batch normalisation, momentum 0.95

	precision	recall	f1-score	support
0	0.86	0.64	0.74	50
1	0.79	0.84	0.81	91
2	0.57	0.43	0.49	68
3	0.88	0.76	0.82	70
4	0.58	0.81	0.68	86
accuracy			0.71	365
macro avg	0.74	0.69	0.71	365
weighted avg	0.73	0.71	0.71	365

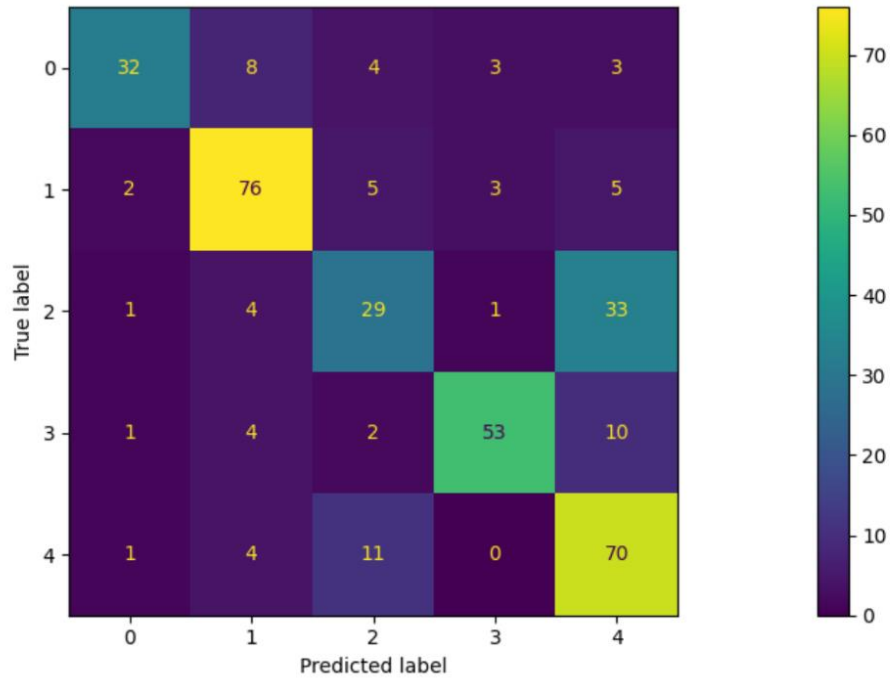

Figure 11. Confusion matrix and classification report for batch normalisation, momentum 0.95

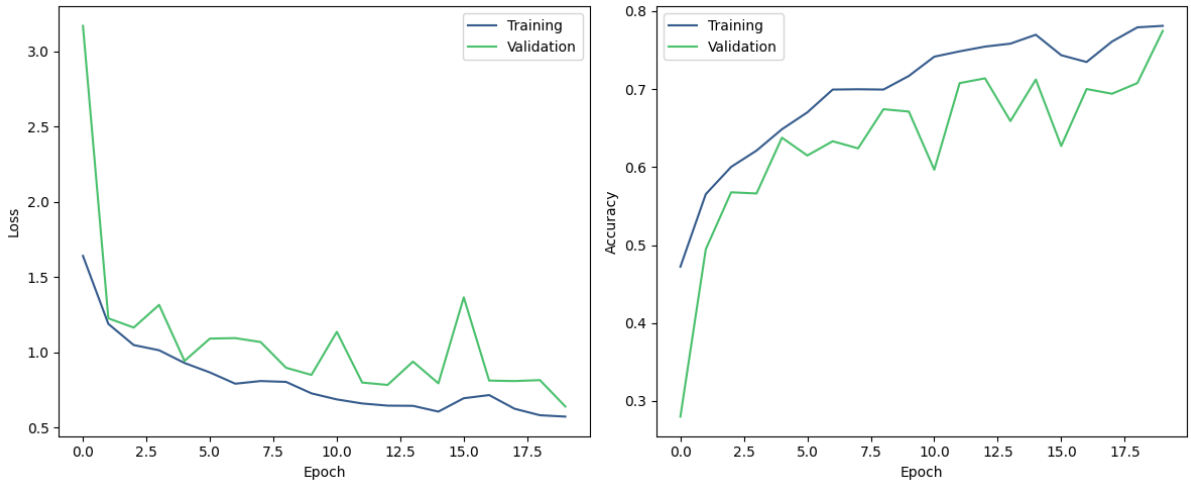

Figure 12. Training and validation accuracy and loss for L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.76	0.84	0.80	50
1	0.77	0.87	0.82	91
2	0.62	0.75	0.68	68
3	0.91	0.74	0.82	70
4	0.81	0.65	0.72	86
accuracy			0.77	365
macro avg	0.78	0.77	0.77	365
weighted avg	0.78	0.77	0.77	365

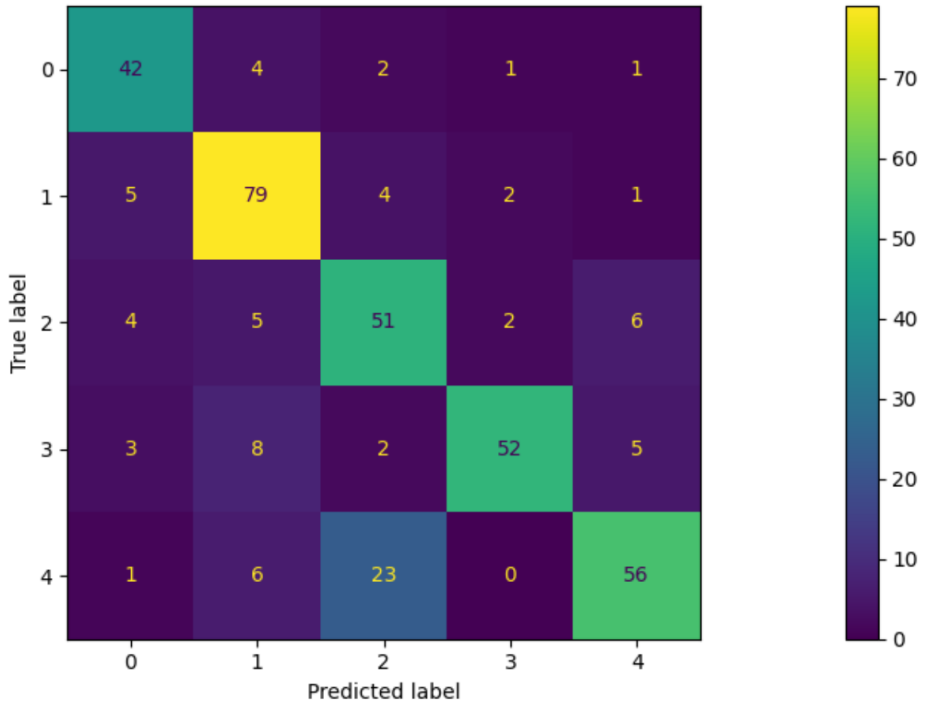

Figure 13. Confusion matrix and classification report for L1 and L2 regularisation (α/β 0.001/0.001)

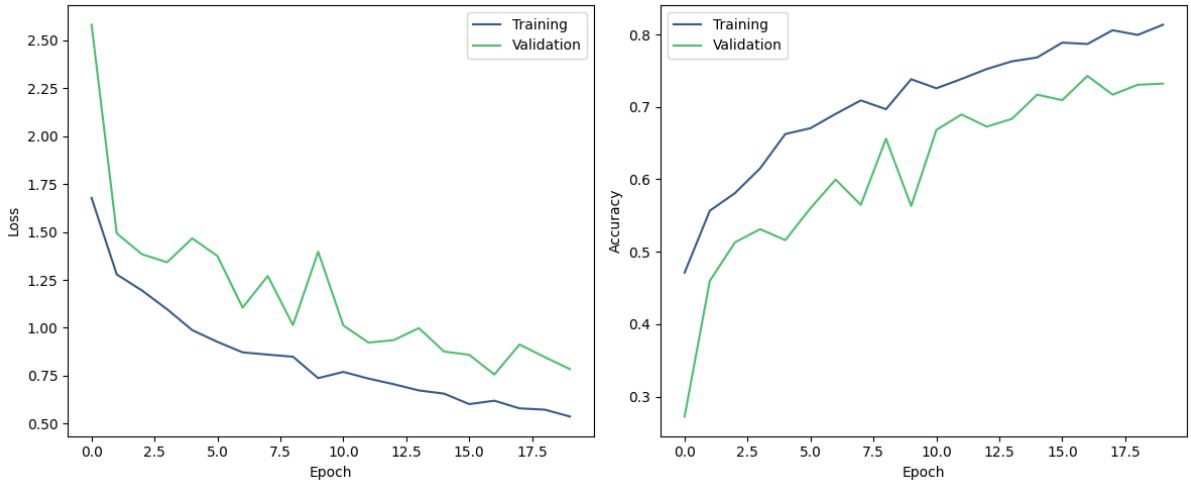

Figure 14. Training and validation accuracy and loss for L1 and L2 regularisation (α/β 0.01/0.01)

	precision	recall	f1-score	support
0	0.89	0.64	0.74	50
1	0.82	0.82	0.82	91
2	0.59	0.78	0.67	68
3	0.83	0.81	0.82	70
4	0.75	0.69	0.72	86
accuracy			0.76	365
macro avg	0.77	0.75	0.75	365
weighted avg	0.77	0.76	0.76	365

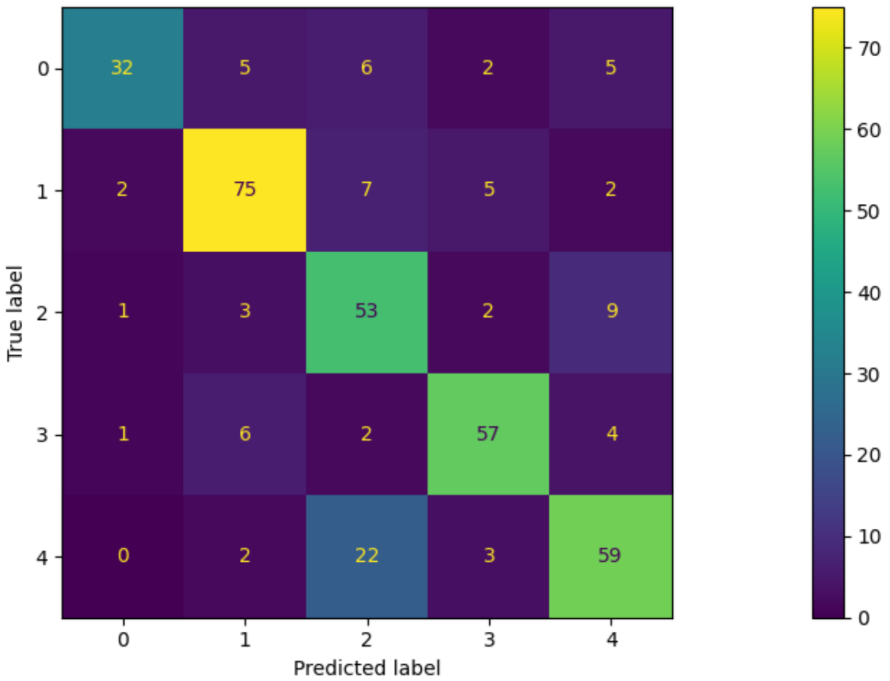

Figure 15. Confusion matrix and classification report for L1 and L2 regularisation (α/β 0.01/0.01)

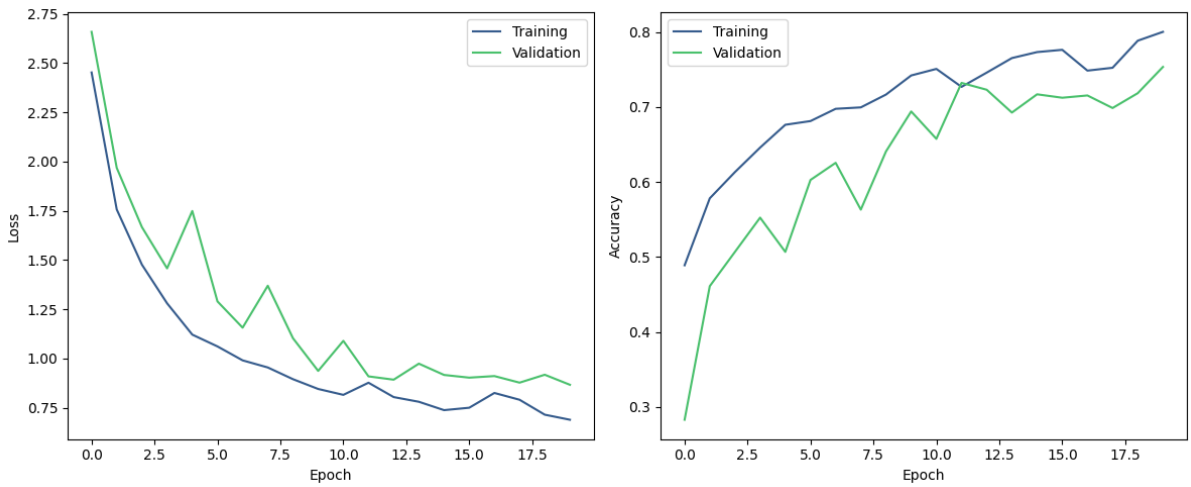

Figure 16. Training and validation accuracy and loss for L1 and L2 regularisation (α/β 0.1/0.1)

	precision	recall	f1-score	support
0	0.66	0.82	0.73	50
1	0.76	0.82	0.79	91
2	0.69	0.75	0.72	68
3	0.88	0.53	0.66	70
4	0.70	0.72	0.71	86
accuracy			0.73	365
macro avg	0.74	0.73	0.72	365
weighted avg	0.74	0.73	0.73	365

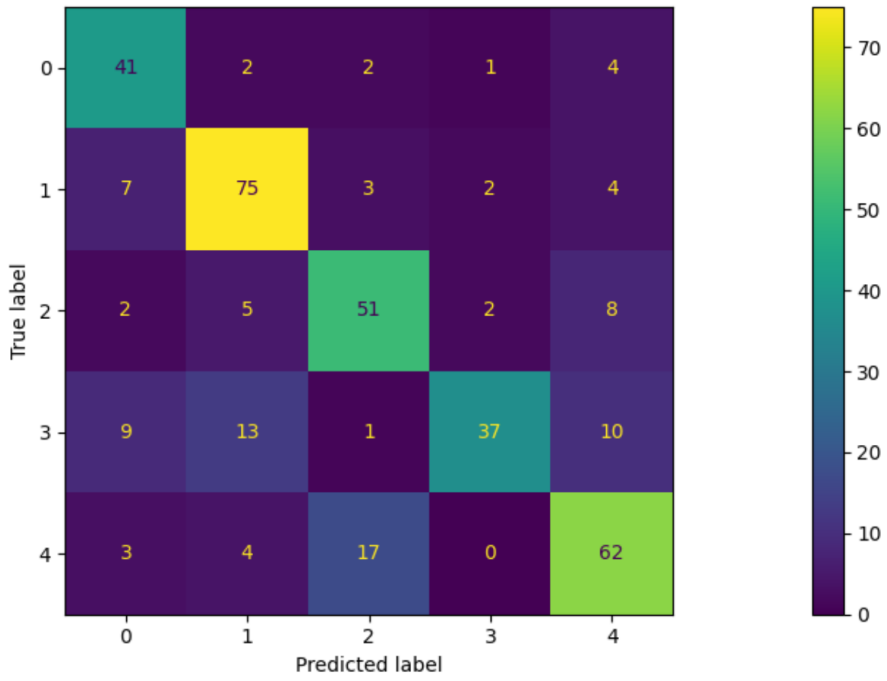

Figure 17. Confusion matrix and classification report for L1 and L2 regularisation (α/β 0.1/0.1)

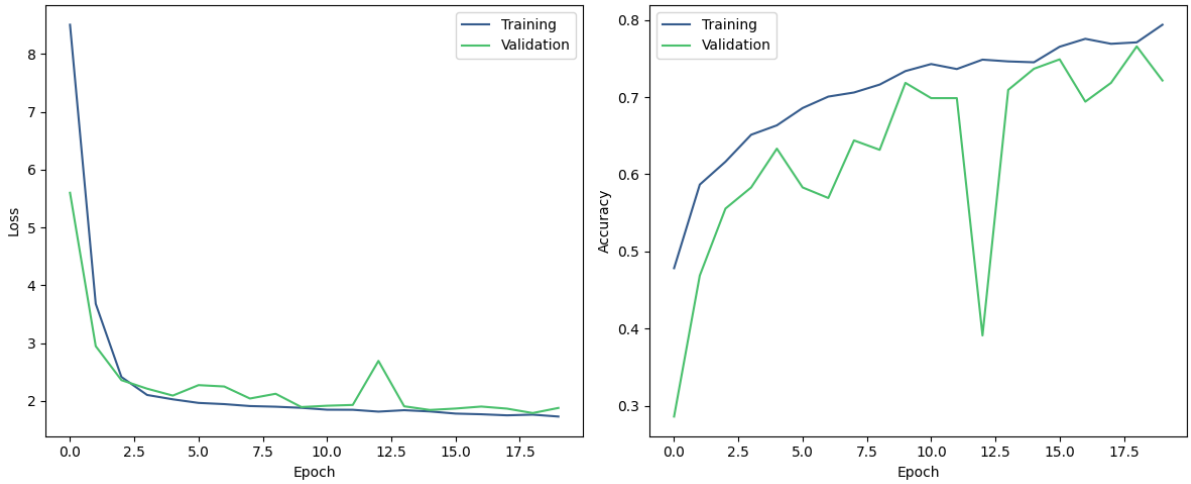

Figure 18. Training and validation accuracy and loss for L1 and L2 regularisation (α/β 1/1)

	precision	recall	f1-score	support
0	0.87	0.78	0.82	50
1	0.73	0.91	0.81	91
2	0.66	0.76	0.71	68
3	0.74	0.89	0.81	70
4	0.93	0.48	0.63	86
accuracy			0.76	365
macro avg	0.79	0.76	0.76	365
weighted avg	0.79	0.76	0.75	365

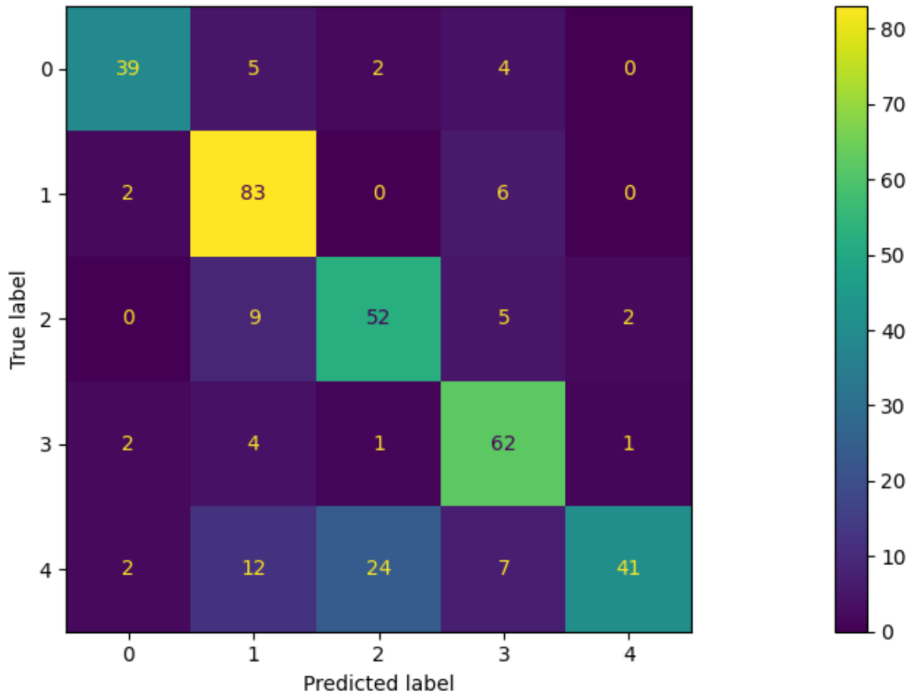

Figure 19. Confusion matrix and classification report for L1 and L2 regularisation (α/β 1/1)

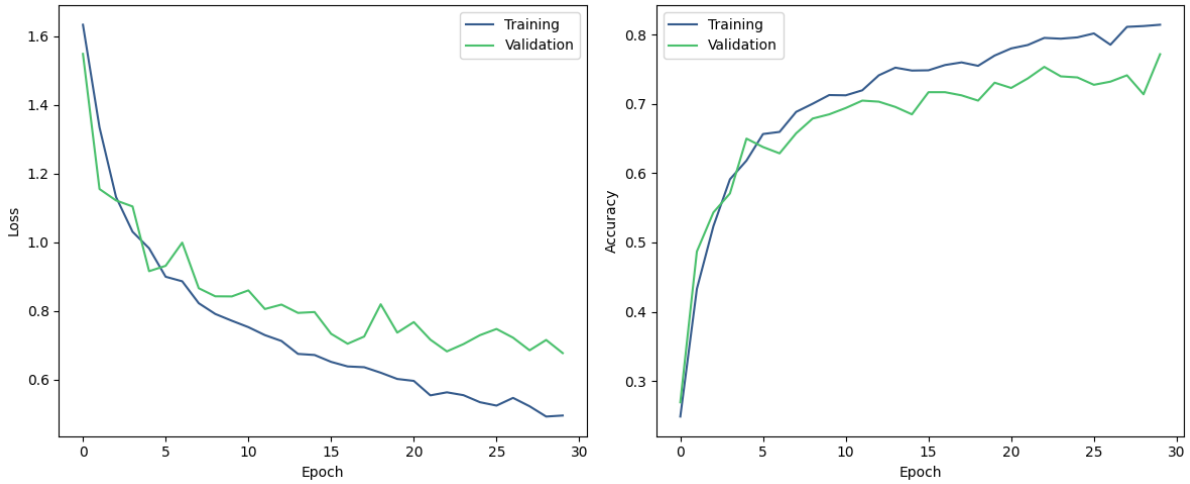

Figure 20. Training and validation accuracy and loss for batch size 32

	precision	recall	f1-score	support
0	0.86	0.86	0.86	50
1	0.84	0.84	0.84	91
2	0.71	0.81	0.75	68
3	0.83	0.74	0.78	70
4	0.76	0.74	0.75	86
accuracy			0.79	365
macro avg	0.80	0.80	0.80	365
weighted avg	0.80	0.79	0.79	365

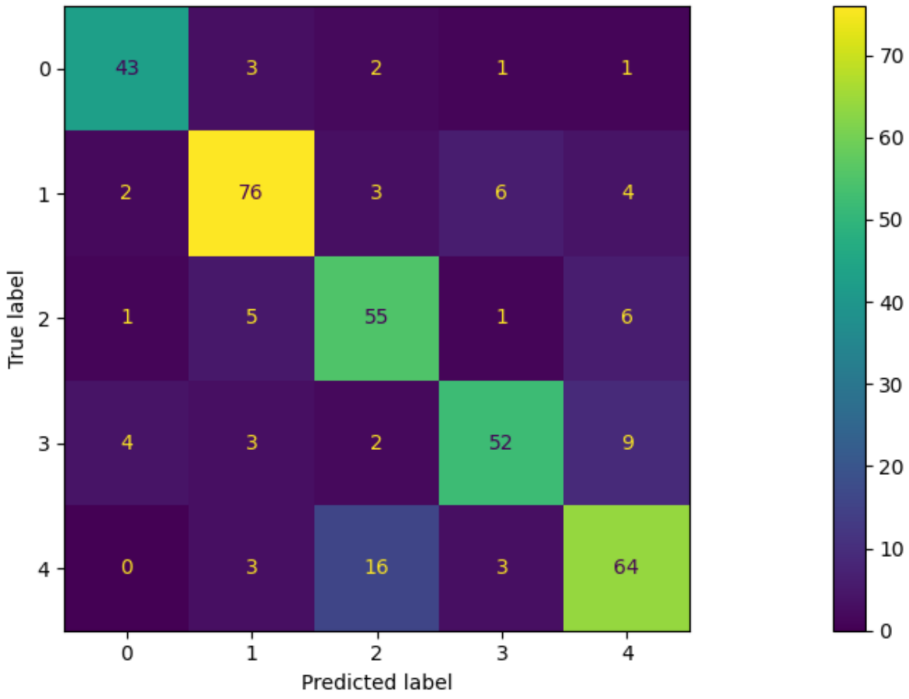

Figure 21. Confusion matrix and classification report for batch size 32

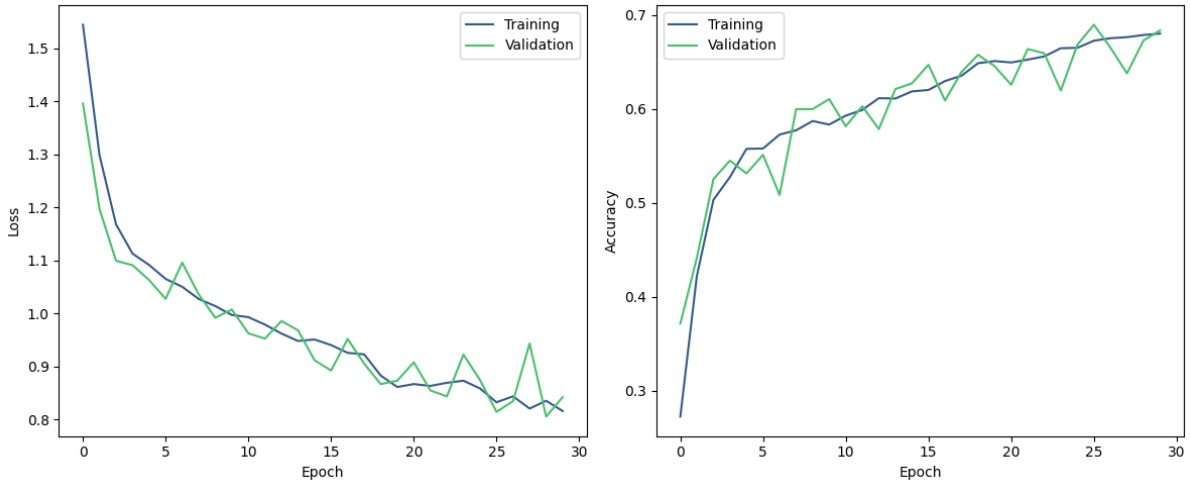

Figure 22. Training and validation accuracy and loss for padding and stride (3,3)

	precision	recall	f1-score	support
0	0.72	0.82	0.77	50
1	0.64	0.79	0.71	91
2	0.53	0.66	0.59	68
3	0.84	0.74	0.79	70
4	0.69	0.40	0.50	86
accuracy			0.67	365
macro avg	0.68	0.68	0.67	365
weighted avg	0.68	0.67	0.66	365

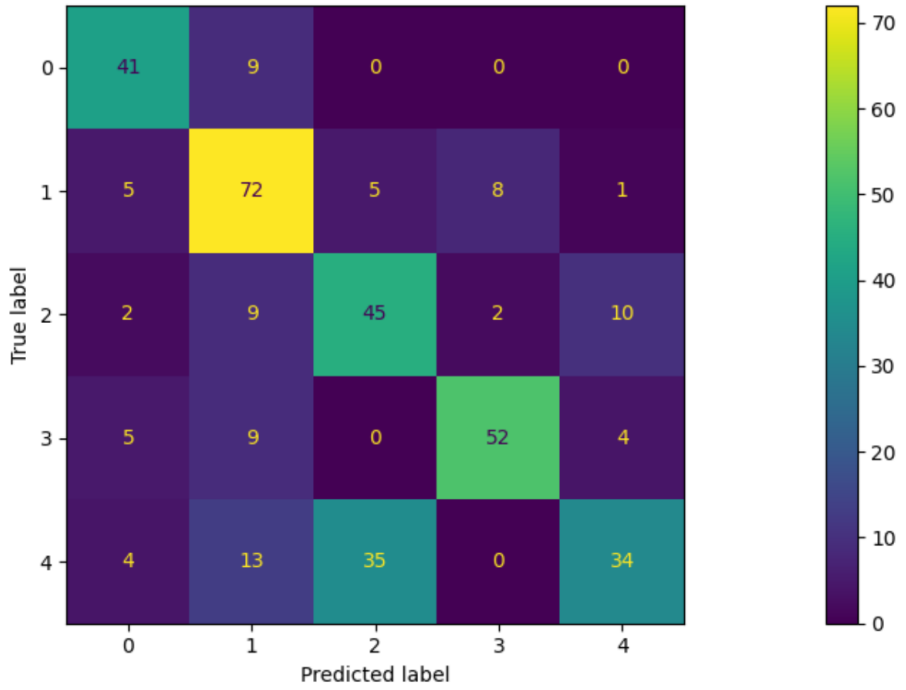

Figure 23. Confusion matrix and classification report for padding and stride (3,3)

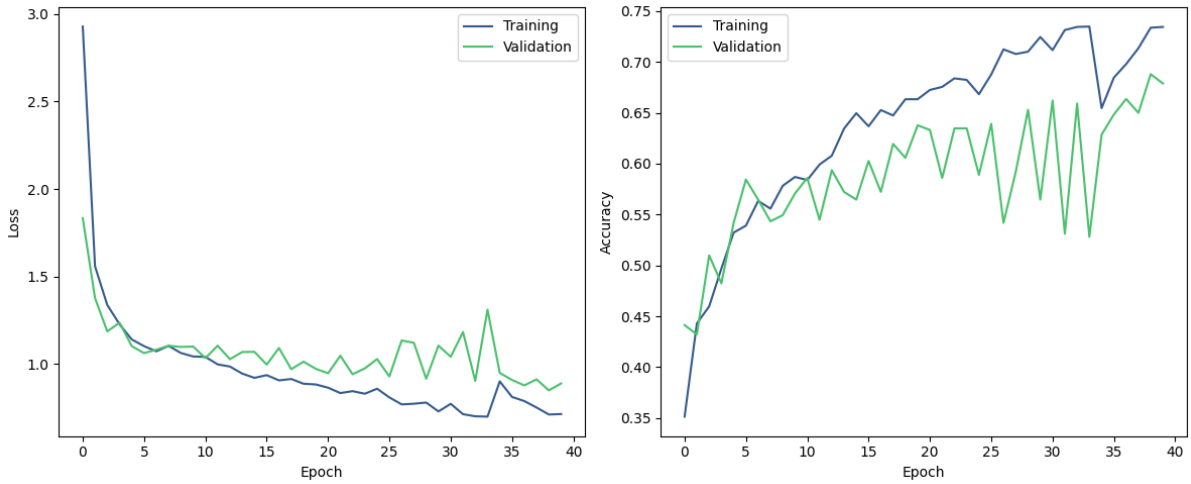

Figure 24. Training and validation accuracy and loss for Adam learning rate 0.01 with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.84	0.72	0.77	50
1	0.63	0.88	0.73	91
2	0.59	0.54	0.56	68
3	0.82	0.90	0.86	70
4	0.73	0.47	0.57	86
accuracy			0.70	365
macro avg	0.72	0.70	0.70	365
weighted avg	0.71	0.70	0.69	365

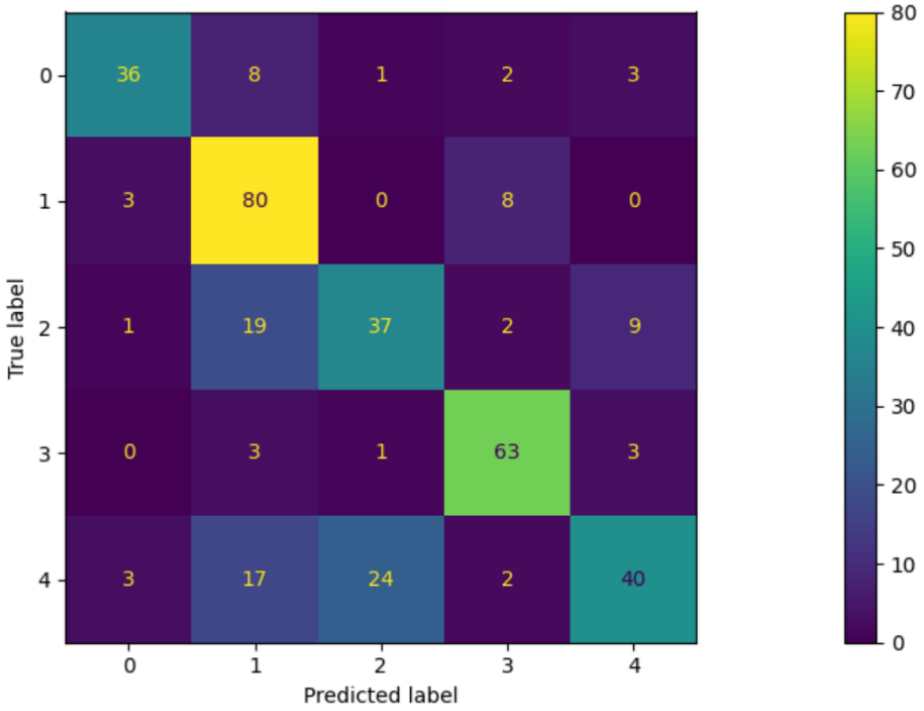

Figure 25. Confusion matrix and classification report for Adam learning rate 0.01 with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

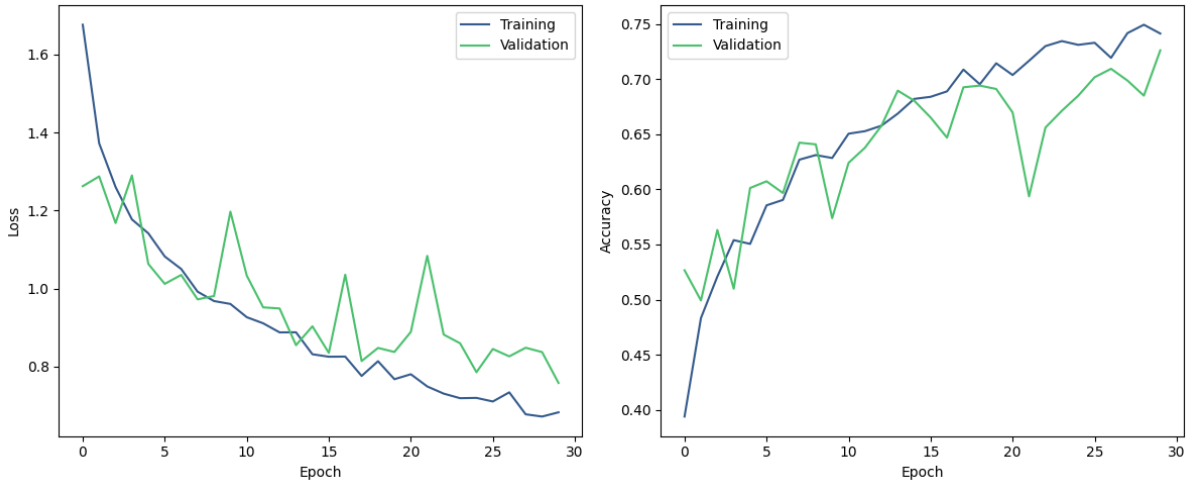

Figure 26. Training and validation accuracy and loss for average pooling with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.74	0.84	0.79	50
1	0.76	0.81	0.79	91
2	0.60	0.68	0.63	68
3	0.79	0.87	0.83	70
4	0.82	0.55	0.66	86
accuracy			0.74	365
macro avg	0.74	0.75	0.74	365
weighted avg	0.75	0.74	0.74	365

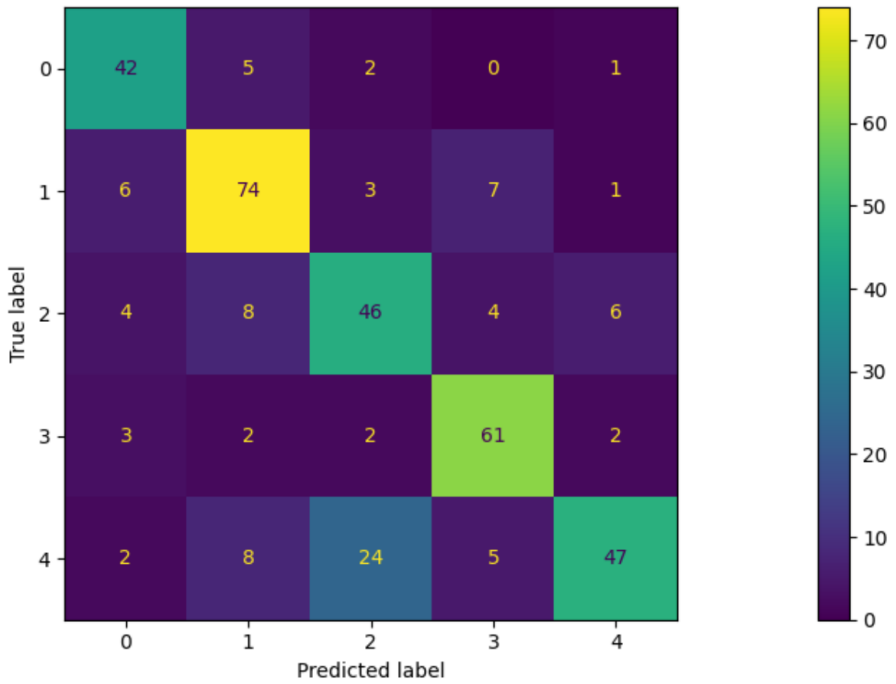

Figure 27. Confusion matrix and classification report for average pooling with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

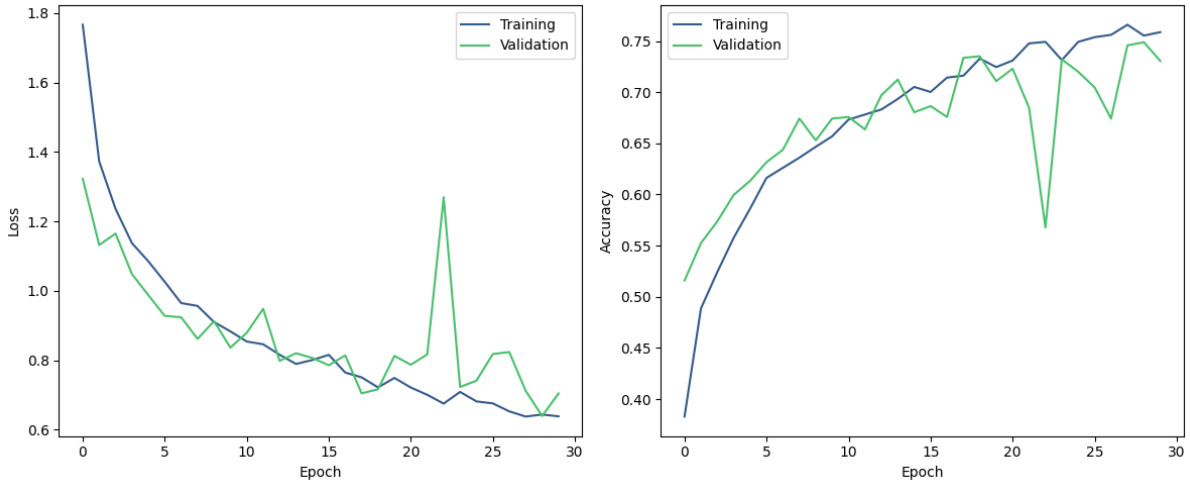

Figure 28. Training and validation accuracy and loss for SeLU activation with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.76	0.84	0.80	50
1	0.68	0.88	0.77	91
2	0.65	0.69	0.67	68
3	0.85	0.83	0.84	70
4	0.89	0.55	0.68	86
accuracy			0.75	365
macro avg	0.77	0.76	0.75	365
weighted avg	0.77	0.75	0.75	365

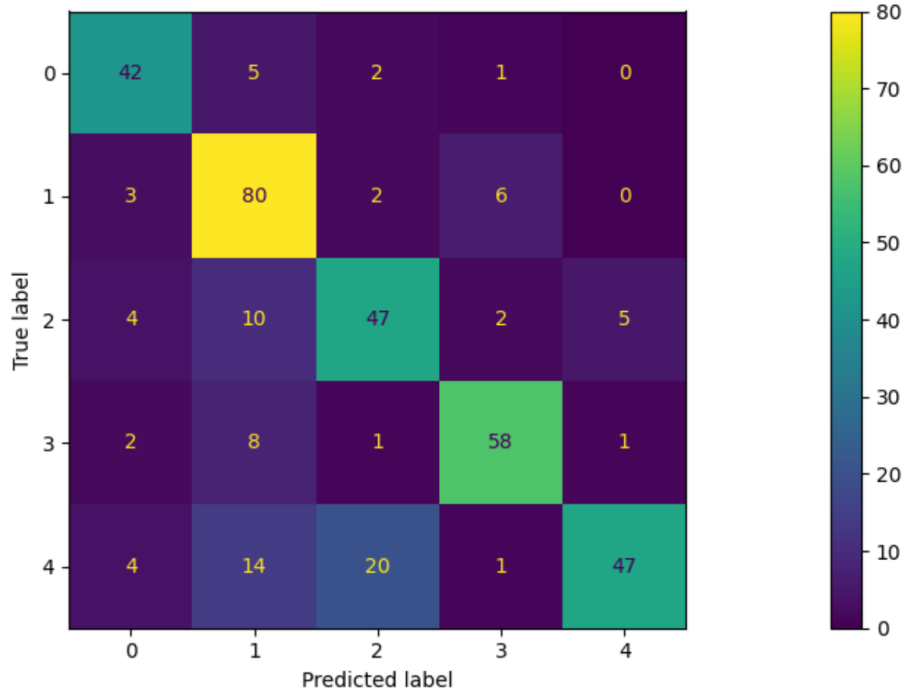

Figure 29. Confusion matrix and classification report for SeLU activation with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

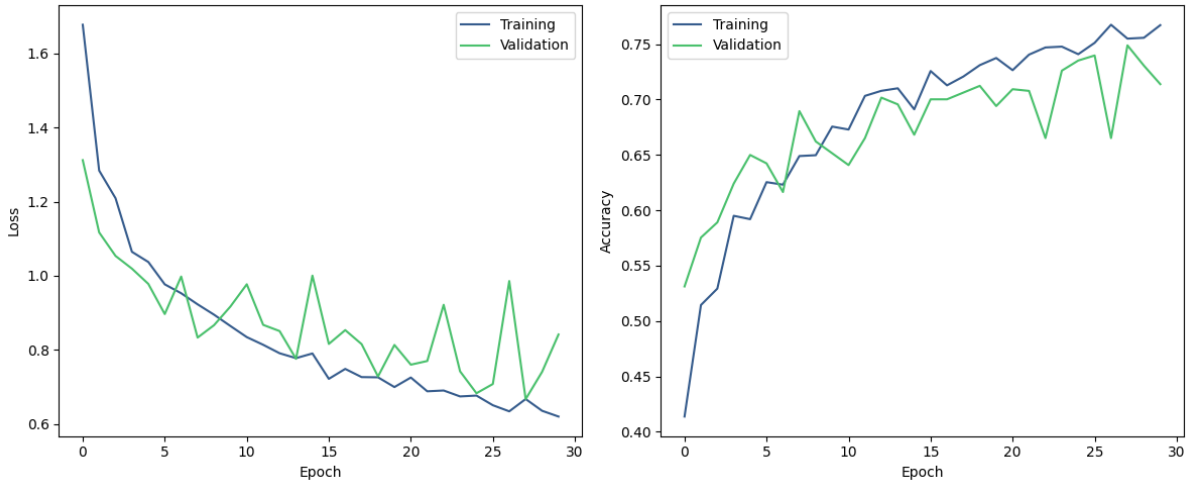

Figure 30. Training and validation accuracy and loss for eLU activation with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.82	0.80	0.81	50
1	0.69	0.84	0.76	91
2	0.59	0.69	0.64	68
3	0.69	0.90	0.78	70
4	0.91	0.37	0.53	86
accuracy			0.71	365
macro avg	0.74	0.72	0.70	365
weighted avg	0.74	0.71	0.69	365

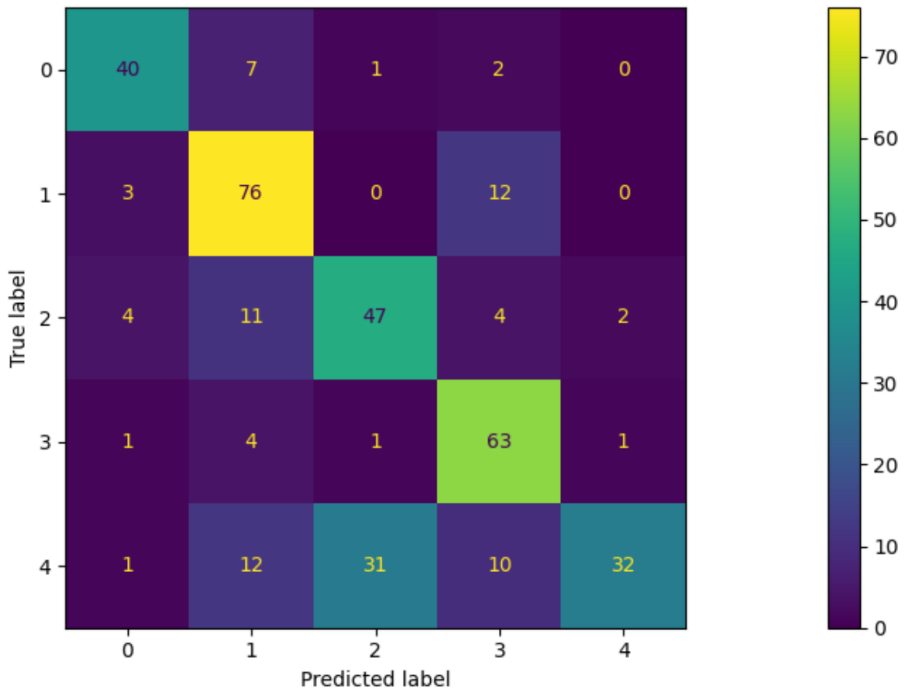

Figure 31. Confusion matrix and classification report for eLU activation with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

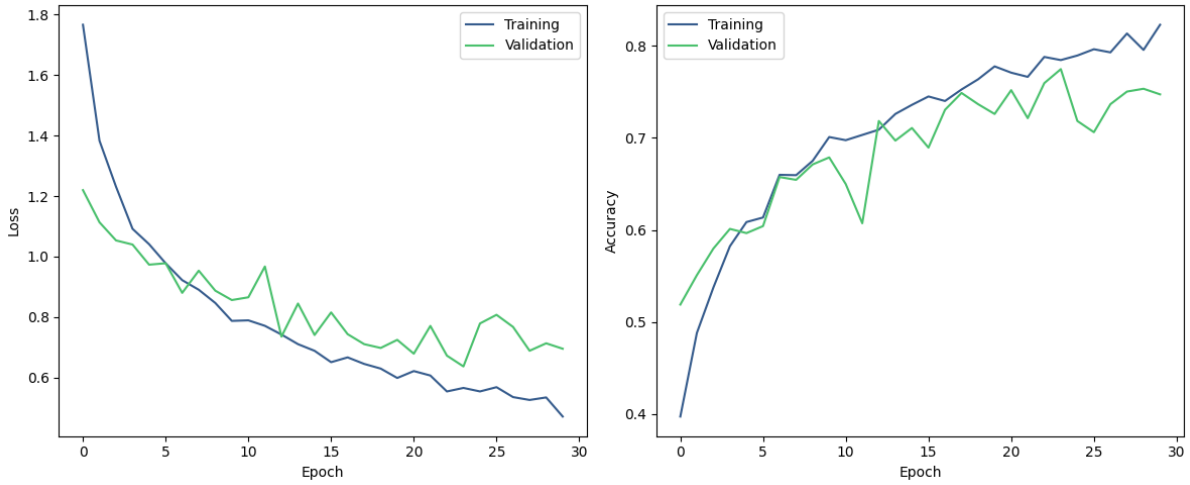

Figure 32. Training and validation accuracy and loss for kernel size (5,5) with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

	precision	recall	f1-score	support
0	0.81	0.84	0.82	50
1	0.80	0.73	0.76	91
2	0.57	0.79	0.66	68
3	0.89	0.79	0.83	70
4	0.79	0.67	0.73	86
accuracy			0.75	365
macro avg	0.77	0.76	0.76	365
weighted avg	0.77	0.75	0.76	365

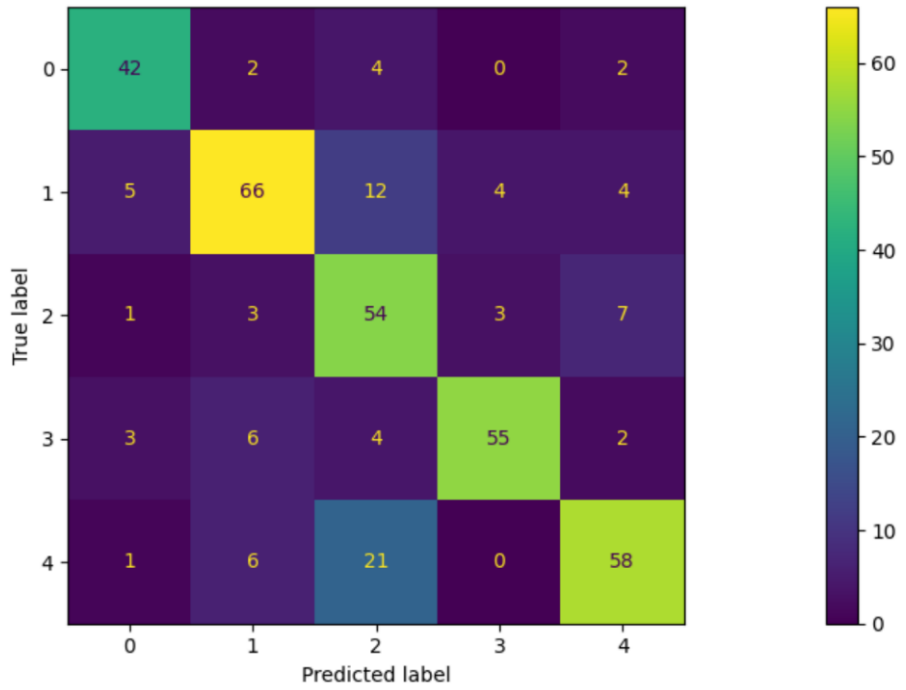

Figure 33. Confusion matrix and classification report for kernel size (5,5) with batch normalisation/L1 and L2 regularisation (α/β 0.001/0.001)

6.2 Tables

Table 1. Regularisation

Iteration	Dropout	Batch normalisation*	L1 and/or L2 (α/β)	Early stopping**
1	0.3	No		No
2	**0.5**	No		No
3	0.3	**Yes**		No
4	0.3	Yes	**L1 and L2 (0.01/0.01)**	No
5	0.3	Yes	**L1 and L2 (0.1/0.1)**	No
6	0.3	Yes	**L1 and L2 (1/1)**	No
7	0.3	Yes	**L1 and L2 (0.001/0.001)**	No
8	**0.2**	Yes	L1 and L2 (0.001/0.001)	No
9	0.3	Yes	L1 and L2 (0.001/0.001)	**Yes**
10	0.3	**No**		Yes
11	**0.5**	Yes	L1 and L2 (0.001/0.001)	No
12	0.3	**Yes (momentum 0.95)**	L1 and L2 (0.001/0.001)	No
13	0.3	**No**	L1 and L2 (0.001/0.001)	No

*Momentum default 0.9, only changed where stated
**Early stopping, patience 7, min_delta 0

Table 2. Hyperparameter tuning

Model	No. Conv2D layers	Adam Optimiser	Pooling	Activation	Kernel	Batch size	Epochs	Padding*	Stride	Regularisation**	Test accuracy (%)	Overfitting	Misclassification***
1	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**1**	75.34	Slight	4- 1,2
2	**4**	0.001	Max	ReLU	(3, 3)	64	20	No	-	1	70.96	Yes	2- 1,4 4- 1
3	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**2**	58.63	Yes	3- 1 4- 1,2
4	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**3**	73.70	Yes	2- 4 4- 2
5	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**4**	75.62	Yes	4- 2
6	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**5**	72.88	Yes	3- 1,4 4- 2
7	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**6**	75.89	Yes	4- 1,2
8	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**7**	76.71	Yes	4- 2
9	5	0.001	Max	ReLU	(3, 3)	64	20	No	-	**8**	70.41	Yes	1- 3 2- 0 4- 2,3

10	5	0.001	Max	ReLU	(3, 3)	64	**40**	No	**-**	**9**	76.44	Yes	4- 2
11	5	0.001	Max	ReLU	(3, 3)	64	40	No	**-**	**10**	76.99	Yes	4- 1,2
12	5	**0.01**	Max	ReLU	(3, 3)	64	40	No	**-**	5	74.79	Yes	4- 2
13	5	0.01	Max	ReLU	(3, 3)	64	40	No	**-**	7	70.14	Yes	2, 4- 1 4- 2
14	5	0.01	Max	ReLU	(3, 3)	64	40	No	**-**	1	74.52	Yes	2, 4- 1
15	5	**0.001**	Max	ReLU	(3, 3)	64	**40**	No	-	1	81.10	Yes	4- 2
16	5	0.001	Max	ReLU	(3, 3)	**32, 128**	40	No	**-**	3	76.16, 79.45	Yes, Yes	4- 2, 4- 1,2,3
17	5	0.001	Max	ReLU	(3, 3)	**32, 128**	40	No	**-**	1	79.45, 78.90	Slight, Yes	4- 2, 4- 2
18	5	0.001	Max	ReLU	(3, 3)	**32**	**30**	**Yes**	**(2, 2)**	1	76.16	Slight	2, 4- 1 4- 2
19	5	0.001	Max	ReLU	(3, 3)	32	30	Yes	(2, 2)	**7**	78.90	Slight	4- 2
20	5	0.001	Max	ReLU	(3, 3)	32	30	Yes	**(3, 3)**	1	66.85	No	4- 2
21	5	0.001	Max	ReLU	(3, 3)	32	30	Yes	(3, 3)	**7**	70.14	Slight	2, 4- 1 4- 2
22	5	0.001	Max	ReLU	**(5, 5)**	32	30	Yes	(2, 2)	7	75.34	Yes	4- 2
23	5	0.001	**Ave**	ReLU	(3, 3)	32	30	Yes	(2, 2)	7	73.97	Slight	4- 2
24	5	0.001	Max	**SeLU**	(3, 3)	32	30	Yes	(2, 2)	7	75.07	Slight	2, 4- 1 4- 2
25	5	0.001	Max	**eLU**	(3, 3)	32	30	Yes	(2, 2)	7	70.68	Yes	1- 3 2- 1 4- 1,2,3
26	5	0.001	Max	ReLU	(3, 3)	32	30	Yes	(2, 2)	**11**	67.12	No	0- 1 1- 3 2- 1,4 4- 1,2
27	5	0.001	Max	ReLU	(3, 3)	32	30	Yes	(2, 2)	**12**	71.23	Slight	2- 4 3- 4
28	5	0.001	Max	ReLU	(3, 3)	32	**60**	Yes	(2, 2)	7	76.44	Yes	4- 2
29	5	0.001	Max	ReLU	(3, 3)	32	60	Yes	(2, 2)	**1**	78.63	Yes	4- 2
30	5	0.001	Max	ReLU	(3, 3)	32	60	Yes	**(3, 3)**	1	77.53	Slight	4- 2 2- 4
31	5	0.001	Max	ReLU	(3, 3)	32	**40**	Yes	(3, 3)	**3**	74.79	Slight	4- 2
32	5	0.001	Max	ReLU	(3, 3)	32	**50**	Yes	(3, 3)	**1**	74.79	Slight	4- 2
33	5	0.001	Max	ReLU	(3, 3)	32	**40**	Yes	**(2, 2)**	**3**	77.81	Yes	4- 1,2 4- 1
34	5	0.001	Max	ReLU	(3, 3)	32	**40**	Yes	**(3, 3)**	**13**	74.25	Slight	2- 1 4-1,2
35	5	0.001	Max	ReLU	(3, 3)	32	**70**	Yes	(3, 3)	**1**	78.08	Slight	2- 4 4- 2

36	5	0.001	Max	ReLU	(3, 3)	32	**40**	Yes	(3, 3)	1	73.70	No	2- 4 4- 2

***Padding was applied to both conv2D and pooling layers.**
**For regularisation methods, see Appendix 6.2, Table 1.
*** 0=daisy, 1=dandelion, 2=rose, 3=sunflower, 4=tulip
Changes made for each model iteration are in bold.
Final optimised model hyperparameters are in red.

6.3 Abbreviations

Convolutional Neural Network	CNN
Exponential linear unit	eLU
Rectified linear unit	ReLU
Scaled exponential linear unit	SeLU
Support Vector Machine	SVM